УДК 512.086

В. А. ШЕХОВЦОВ, канд. техн. наук, НТУ «ХПИ»

ОБРАБОТКА ТРЕБОВАНИЙ КАЧЕСТВА НА РАННИХ СТАДИЯХ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

У статті розглянуто підхід до аналізу вимог якості на перших стадіях розробки програмного забезпечення. Підхід базується на моделі Клагенфуртського концептуального предпроектування (KCPM), розширюючи її для обробки семантики вимог якості. Ці вимоги трактуються як наскрізні інтереси. Представлені розширення моделі KCPM та нові правила відображення при використанні асиметричного аспектно-орієнтованого підходу.

В статье рассматривается подход к анализу требований качества на первых этапах разработки программного обеспечения. Подход основан на модели Клагенфуртского концептуального предпроектирования (KCPM), расширяя ее для обработки семантики требований качества. Эти требования трактуются как сквозные интересы. Представлены расширения модели KCPM и новые правила отображения при использовании асимметричного аспектно-ориентированного подхода.

This paper presents an approach for processing the quality requirements on early development stages. It is based on Klagenfurt Conceptual Predesign Model (KCPM) allowing taking into account the semantics of quality requirements. These requirements are treated as crosscutting concerns. The extensions of the KCPM schema and the new mapping rules are introduced for the case of asymmetric aspect-oriented approach.

1. Введение. В настоящее время при разработке программного обеспечения всё более актуальной становится проблема корректного учета требований качества, под которыми понимаются требования к таким характеристикам системы, как производительность, надёжность, эффективность, применимость, переносимость, тестируемость, воспринимаемость, модифицируемость и т.д. [1]. Это обусловлено увеличивающейся сложностью создаваемых систем, временными ограничениями и повышением требований к самим программным продуктам вследствие быстроты изменения окружающего мира. Проблема определения и обработки требований качества достаточно сложна и не до конца исследована. Некоторые работы (в частности, [2-4]) содержат попытки описать подходы к решению данной проблемы, но вопрос об универсальной концепции обработки таких требований остается открытым.

В данной статье будет описана технология преобразования требований качества к программной системе в промежуточную семантическую модель с последующим построением на её основе концептуальной модели системы. В качестве основы для промежуточной модели используется модель Клагенфуртского концептуального предпроектирования (Klagenfurt Conceptual Predesign Model, KCPM) [5-7].

Дальнейшее изложение будет организовано следующим образом. В разделе 2 будет приведено описание основных используемых концепций, раздел 3 посвящен постановке задачи, раздел 4 – описанию предложенного подхода,

в заключительном разделе приводятся выводы и программа дальнейших исследований.

2. Предварительная информация. В данном разделе мы приведем основные сведения об используемых нами базовых подходах - аспектно-ориентированной разработке программного обеспечения (АОРПО) и Клагенфуртском концептуальном предпроектировании.

2.1. Аспектно-ориентированная разработка программного обеспечения. Основной целью методологии АОРПО является реализация средств систематической идентификации, разделения, представления и компоновки сквозных интересов на различных этапах разработки программного обеспечения. *Интерес* представляет собой цель или концепцию, которая реализована в компоненте системы.

Основным принципом АОРПО является *разделение интересов* [8]. При этом программное обеспечение рассматривается как реализация сложного взаимодействия различных базовых и системных интересов. Базовые интересы связаны с основной функциональностью системы, системные интересы – с обеспечением взаимодействия между компонентами системы. Например, для системы обеспечения продаж основными интересами могут быть управление прайс-листами и обработка заказов. Системные интересы в данном случае могут включать протоколирование операций, обеспечение целостности и непротиворечивости данных, резервное копирование информации.

Разделение интересов означает, что различные виды или типы интересов должны быть идентифицированы и разграничены с тем, чтобы понизить сложность разрабатываемой системы. Это разделение может проводиться на различных этапах разработки программного обеспечения - при анализе требований, проектировании и реализации. Наиболее приемлемым решением является осуществление этого процесса на всех стадиях разработки – от анализа требований к реализации.

Системные интересы, как правило, затрагивают различные компоненты системы, принято говорить, что такие интересы отражают *сквозное поведение* – поведение, которое присутствует во многих компонентах системы, возможно, не связанных друг с другом. Так, например, протоколирование (сохранение информации в системном журнале) должно быть реализовано в любом компоненте системы, поведение которого необходимо отслеживать. Интересы, связанные со сквозным поведением, называются сквозными интересами или аспектами. Сквозное поведение обладает рядом свойств: его реализация не является основной задачей компонентов системы; схема его организации не зависит от основной функциональности компонентов; такое поведение требуется во многих классах компонентов и во многих методах этих классов.

С реализацией сквозных интересов системы связан ряд проблем. Рассмотрим их симптомы:

1. *Запутанность кода.* Если код содержит фрагменты реализации нескольких интересов, разобраться в его организации становится сложно.

2. *Разбросанность кода.* Невозможно выделить отдельный модуль, реализующий сквозной интерес – соответствующая реализация распределена по многим модулям системы.

Выделяют два основных класса подходов к реализации систем на основе АОРПО: асимметричные и симметричные подходы. Основным принципом асимметричных подходов является то, что базовые и сквозные интересы выделяются отдельно и реализуются различным образом. При этом правила компоновки интересов задаются внутри реализации сквозных интересов (аспектов). Наиболее известным подходом этого класса является аспектно-ориентированное программирование (АОП) [9].

Основным принципом симметричных подходов является то, что все интересы рассматриваются и реализуются единообразно. Правила компоновки интересов задаются отдельно от их реализации. Наиболее известным подходом этого класса является многомерное разделение интересов (МРИ) [10].

АОП реализует чёткое разделение между базовыми классами и аспектами, при этом как те, так и другие собираются в конечную систему в ходе процесса композиции аспектов. Рассмотрим основные конструкции АОП, которые мы будем использовать в дальнейшем:

1. Аспекты (aspects) – модули, реализующие сквозное поведение. При этом каждый из аспектов разрабатывается отдельно, а на конечном этапе для получения завершенной системы производится композиция аспектов.
2. Точки соединения (join points) – точно определённые позиции в базовых классах (вызов метода, доступ к свойству и т.д.), в которых их выполнение может быть прервано для запуска сквозного кода. Модель точек соединения определяет допустимое множество таких позиций.
3. Срезы (pointcuts) – правила, определяющие набор точек соединения. Они определяют места, где сквозной код перехватывает выполнение базовых классов. Например, срез может охватывать точки вызова всех методов некоторого класса.
4. Уведомления (advices) – блоки сквозного кода аспекта, которые запускаются в точках соединения, определённых срезом. Примером уведомления может служить код записи в системный журнал, выполняемый перед всеми вызовами методов класса. На стадии композиции код уведомления внедряется в код базовых классов в местах, определённых срезом.

Наиболее известной реализацией АОП является AspectJ [11]. Это – среда программирования, поддерживающая расширение языка Java, в котором определены конструкции для описания сквозного поведения и средства композиции аспектов.

2.2. Модель Клагенфуртского концептуального предпроектирования. Традиционный процесс разработки программного обеспечения на начальном этапе включает в себя стадии анализа требований и концептуального

моделирования или проектирования. При этом происходит резкий переход от первой стадии ко второй. Подразумевается, что информация о предметной области может быть легко извлечена из спецификаций требований, после чего на ее основе легко построить концептуальную модель системы. Практика, однако, показывает, что реализация такого подхода достаточно сложна и приводит к многочисленным ошибкам. Причиной является несоответствие спецификаций требований и концептуальной модели. Одним из подходов к решению данной проблемы является Клагенфуртское концептуальное предпроектирование [5-7]. Основной целью этого подхода является реализация такой процедуры сбора и анализа требований, которая допускает всесторонний контроль со стороны заказчика программного обеспечения.

Данная цель достигается с помощью модели Клагенфуртского концептуального предпроектирования (KCPM) – промежуточной семантической модели, которая строится после стадии сбора требований, но перед построением концептуальной модели. Данная модель содержит семантические понятия, которые обладают большей общностью по сравнению с понятиями концептуального моделирования. Они более понятны и доступны конечному пользователю (заказчику) и могут быть легко проверены им. В качестве таких понятий выступают, в частности:

1. Тип-вещь (thing-type) – обобщение понятий объекта, класса и атрибута.
2. Тип-связь (connection-type) – обобщение связей в системе.
3. Тип-операция (operation-type) – обобщение действий, выполняемых в системе.

Данная модель может быть представлена в табличной (шаблонной) форме или в виде семантической сети. В дальнейшем мы будем использовать шаблонное представление. После проверки пользователем, модель предпроектирования может быть преобразована в концептуальную модель. Это преобразование осуществляется с помощью специальных правил. Мы рассмотрим шаблонное представление и правила преобразования в следующих разделах.

Данная модель может быть построена на основе спецификации требований, при этом стандартная ее реализация [5-7] позволяет использовать для этого исключительно функциональные требования. Существует два способа построения модели: непосредственно аналитиком и из текста спецификации с использованием обработки естественного языка. Используя второй подход, возможно построение концептуальной модели на основании текста спецификации. Этому посвящён проект NIBA [12], частью которого является проект по разработке KCPM.

3. Постановка задачи. Проанализировав текущую ситуацию, можно видеть, что при обработке требований качества на начальных этапах разработки программного обеспечения возникает несколько проблем, требующих своего решения:

1. Модель концептуального предпроектирования на настоящий момент реализует сбор семантики и преобразование функциональных требований [7]. Для поддержки обработки требований качества данную модель необходимо модифицировать.
2. Процесс аспектно-ориентированной разработки программного обеспечения не содержит этапа, аналогичного по своей функциональной нагрузке концептуальному предпроектированию (для него не определена промежуточная семантическая модель).

В данном случае представляется целесообразным совместить преимущества АОРПО с возможностями, предоставляемыми KCPM. Только в этом случае можно получить законченную технологию сбора семантики и преобразования требований качества. Целью данной статьи является освещение первых шагов при создании подхода, позволяющего решить эту задачу.

4. Аспектное предпроектирование. Рассмотрим вопрос представления и преобразования требований качества с использованием расширенной модели предпроектирования. Для обработки этих требований будем следовать аспектному подходу, который определяет их как сквозные интересы (аспекты) системы. Для обозначения происхождения требований, с которыми предполагается работать, предложенный подход назван *аспектным предпроектированием* [13, 14].

Как описано в работе [5], процесс предпроектирования, не учитывающий требования качества, состоит из двух этапов:

1. Заполнение обобщённой KCPM-схемы (набора шаблонов) на основе требований и верификация информации данной схемы пользователем (заинтересованным лицом).
2. Осуществление преобразования данных из KCPM-схемы в концептуальную модель.

Для учета необходимости представления требований качества необходимо произвести определенные изменения этих этапов.

4.1. Расширение модели концептуального предпроектирования. Основной проблемой аспектного предпроектирования является необходимость определения семантических понятий, которые бы адекватно представляли сквозные интересы в системе. При этом эти понятия должны оставаться доступными для конечного пользователя, который осуществляет проверку данных.

Одновременно удовлетворить двум этим требованиям достаточно непросто. Главной причиной является терминология АОРПО, которая представляется сложной для восприятия конечным пользователем. Особенно это касается использования таких понятий как аспекты, точки соединения, срезы, уведомления и т.д. Данный факт вызван следующими причинами:

1. АОП было создано профессиональными программистами, проблемы которых в значительной степени отличаются от проблем обычных пользователей.

2. Полностью разобраться в технологии сквозных интересов достаточно сложно даже для опытных разработчиков и аналитиков.

Проблема сложной терминологии распространяется не только на конечных пользователей, но и на аналитиков. Например, для того чтобы промоделировать точки соединения с помощью UML, необходимо изменить их определение в терминологии AspectJ “принципиальные точки выполнения программы” [11] на более общее - ”места для добавления расширений”. В данной статье мы рассмотрим первые шаги для решения описанной проблемы. Наш подход является упрощённым, так как мы не рассматриваем сложные определения срезов, представление семантики таких определений является темой для дальнейших исследований.

Перед тем, как перейти к описанию подхода, нам необходимо привести пример спецификации требований, который будет использоваться при дальнейшем изложении. Рассмотрим систему поддержки работы банка. Фрагмент описания требований к данной системе выглядит следующим образом: *Система поддержки работы банка работает с клиентами и счетами (П1). Все операции со счетами необходимо протоколировать (П2). При попытке клиента снять средства со счёта, система обязана запросить у него пароль (П3).*

4.2. Моделирование семантики интересов. Семантическим понятием, представляющим сквозной интерес, является тип-вещь. Для того, чтобы отличать аспектные типы-вещи от стандартных, колонка шаблона «классификация» для них будет содержать значение «интерес». Данное значение устанавливает проектировщик. Пример шаблона, учитывающего семантику сквозных интересов, представлен на рис. 1. В предметной области приведенного выше примера в качестве таких интересов выступают *Безопасность* и *Протоколирование*.

Предметная область: банковская система				
Код#	**Название**	**Классификация**		**Источник**
B001	Безопасность	интерес		П3
B002	Протоколирование	интерес		П2
B003	Счёт	тип-вещь		П1, П2, П3
B004	Клиент	тип-вещь		П1, П3
...				

Рисунок 1 – Фрагмент шаблона моделирования семантики сквозных интересов

4.3. Моделирование семантики уведомлений. Семантика уведомления достаточно сходна с семантикой типа-операции, основное отличие заключается в механизме вызова, так как уведомление вызывается неявно в точках соединения, связанных с ним посредством среза. В нашей модели мы определяем следующее:

1. Тип-вещь, на который ссылается уведомление, является аспектом;
2. Колонка шаблона для описания вида вызова должна содержать значение «автовызов».

Для обеспечения возможности вызова уведомлений необходима дополнительная информация, обеспечиваемая срезами, которая будет описана ниже. Рис. 2 содержит фрагмент шаблона для типов-операций с учётом семантики уведомлений, представляющий приведенный выше пример спецификации требований.

Предметная область: банковская система					
Код#	**Название**	**Вид вызова**	**…**	**Выполняющий тип**	**Источник**
O001	Снятие средств	явный		B004	П3
O002	Запрос пароля	автовызов		B001	П3
O003	Протоколирование	автовызов		B002	П2
…					

Рисунок 2 – Фрагмент шаблона моделирования семантики уведомлений

4.4. Моделирование семантики точек соединения. Семантика большинства точек соединения в спецификациях требований относится к двум категориям:

1. Вызов типа-операции.
2. Доступ к типу-вещи.

Соответствующие категории в АОП, применяемые на следующих этапах разработки, также могут иметь две различные реализации:

1. Доступ к соответствующему полю или классу.
2. Вызов всех методов соответствующего класса.

Таким образом, можно утверждать следующее: в большинстве случаев точки соединения могут быть представлены или типом-операцией, или типом-вещью. Далее мы опишем способ применения такого представления.

4.5. Моделирование семантики срезов. Срез представляет собой связь между уведомлением и точкой соединения, таким образом, он может быть определён как тип-связь. Чтобы позволить типу-связи представлять срез, необходимо изменить структуру шаблона для него. Нам необходимо обеспечить для среза возможность ссылаться не только на типы-вещи, но и на уведомления, с которыми он связан. Для этого необходимо разделить колонку «связанный тип-вещь» [7] на две:

1. «Связанный тип» - код типа, на который ссылается тип-связь.
2. «Связанная концепция» - обозначение общей семантической концепции для связанного понятия («тип-операция» или «тип-вещь»).

На рис. 3 представлен фрагмент шаблона типов-связей с учетом семантики срезов, соответствующий нашему примеру спецификации требований.

Срез С001 соединяет уведомление для запроса пароля с вызовом операции снятия средств. Срез С002 соединяет уведомление протоколирования со всеми операциями типа-вещи *Счет*.

Предметная область: банковская система							
Код#	**Название**		**Перспективы**				**Источник**
			Код перспект.	**Связан. тип**	**Связан. концепция**	**Название**	
С001	Пароль для операции		ПС001-1	О002	тип-операция	Запрос пароля	П3
			ПС001-2	О001	тип-операция	Снятие средств	
С002	Протоколир. всех операций		ПС002-1	О003	тип-операция	Протоколирование	П2
			ПС002-2	В003	тип-вещь	Счёт	
...							

Рисунок 3 – Фрагмент шаблона моделирования семантики срезов

4.6. Расширение процесса преобразования. Для описания процесса преобразования необходимо выбрать конечную модель или нотацию. В данной работе мы рассмотрим преобразование модели аспектного предпроектирования в набор стандартных понятий асимметричного АОРПО-подхода, имеющих прямое соответствие в таких системах, как AspectJ. (аспект, уведомление, срез и точка соединения). Такой выбор позволяет добиться определенной общности решения, т.к. многие модели концептуального представления сквозных интересов предлагают свое соответствие каждому из этих понятий, поэтому для преобразования описанной модели в модель одного из этих подходов достаточно привести результат применения рассмотренных ниже правил в соответствие этому представлению.

Преобразование сводится к применению следующих правил:

1. *Правило аспекта*. Тип-вещь T преобразуется в аспект A_T в случае задания аналитиком для T значения «интерес» в поле «Классификация». После применения данного правила, все операции, связанные с данным типом-вещью, будут преобразованы в уведомления.
2. *Правило уведомления*. Тип-операция O преобразуется в уведомление AD_O, если O ссылается на тип-вещь, преобразованный в аспект и в колонке «вид вызова» содержит значение «автовызов».
3. *Правило среза*. Тип-связь C преобразуется в срез P_C, если C ссылается на тип-операцию, уже преобразованный в уведомление.
4. *Правило точки соединения № 1*. Тип-операция O преобразуется в точку соединения J_O, если O уже преобразован в метод M_O и связан с типом-связью, уже преобразованным в срез. В этом случае J_O представляет собой вызов метода M_O.

5. *Правило точки соединения № 2.* Тип-вещь T преобразуется в точку соединения J_T, если T уже преобразован в тип-значение V_T и связан с типом-связью, уже преобразованным в срез. В этом случае J_T представляет собой доступ к этому типу-значению (атрибуту).
6. *Правило точки соединения № 3.* Тип-вещь T преобразуется в точку соединения J_T, если T уже преобразован в класс C и связан с типом-связью, уже преобразованным в срез. В этом случае J_T представляет собой вызов любого метода класса C.

Для приведенного примера нужно применить следующие правила:

1. Правило аспекта для преобразования типов-вещей *Безопасность* (B001) и *Протоколирование* (B002) в аспекты.
2. Правило уведомления для преобразования типов-операций *Запрос пароля* (O002) и *Протоколирование* (O003) в уведомления.
3. Правило среза для преобразования типов-связей C001, C002 в срезы.
4. Правила точек соединения 1 и 3 для преобразования типа-операции *Снятие средств* (O001) и типа-вещи *Счёт* (B003) в точки соединения.

5. Заключение. В данной статье описаны первые шаги в направлении решения проблемы обработки требований качества на начальных стадиях разработки программного обеспечения. Для проверки применимости идей данной статьи необходимы реальные проекты и пользователи. Только после проверки в реальных условиях - как модели, так и правил преобразования - можно будет утверждать о законченности данного подхода.

Открытым также является вопрос об интеграции рассмотренного подхода в общий процесс управления качеством разработки программного обеспечения. Мы предполагаем, что семантическая модель, подобная рассмотренной в данной работе, может стать узловой для такого процесса.

Список литературы: ***1.*** ISO/IEC 9126-1, Software Engineering – Product Quality – Part 1: Quality model, 2001. ***2.*** *Chung, L., Nixon, B., et al.* Non-Functional Requirements in Software Engineering. Kluwer, 2000. ***3.*** *Mylopoulos, J., Chung, L., Nixon, B.* Representing and Using Non-Functional Requirements: A Process-Oriented Approach. // IEEE Transactions on Software Engineering, Vol. 18, No. 6, pp. 483-497. ***4.*** *Glinz, M.* On Non-Functional Requirements. // Proc.RE'07, IEEE, 2007. ***5.*** *Mayr, H.C., Kop, Ch.* A User Centered Approach to Requirements Modeling. // Proc. Modellierung 2002. LNI P-12, GI-Edition, 2002, p. 75-86. ***6.*** *Kop, Ch., Mayr, H.C., Zavinska, T.* Using KCPM for Defining and Integrating Domain Ontologies. // WISE 2004 Workshops, LNCS 3307, Springer, 2004, p. 190-200. ***7.*** *Kop, Ch., Mayr, H.C.* Mapping Functional Requirements: From Natural Language to Conceptual Schemata. // Proc. SEA'02, 2002, p. 82-87. ***8.*** *Дейкстра, Э.* Дисциплина программирования. М.: Мир, 1978. ***9.*** *Kiczales, G., Lamping, J., et al.* Aspect-Oriented Programming. // Proc. ECOOP'97, 1997. ***10.*** *Tarr, P., Ossher, H., et al.* N Degrees of Separation: Multi-Dimensional Separation of Concerns. // Proc. ICSE'99, 1999. ***11.*** *Laddad, R.* AspectJ in Action. Manning. 2005. ***12.*** *Niba, L.C.* The NIBA workflow: From textual requirements specifications to UML schemata. // Proc. ICSSEA'2002, Paris, 2002. ***13.*** *Shekhovtsov, V., Kostanyan, A.* Aspectual Predesign. // Proc. ISTA'2005, LNI P-63, GI-Edition, 2005, p. 216-226. ***14.*** *Shekhovtsov, V., Kostanyan, A., Gritskov, E., Litvinenko, Y.* Tool Supported Aspectual Predesign // Proc. ISTA'2006, LNI P-84, GI-Edition, 2006, p. 153-164.

Поступила в редколлегию 17.11.07